Beyond Snapping: Persistent, Tweakable Alignment and Distribution with StickyLines

Marianela Ciolfi Felice[1,2] **Nolwenn Maudet**[1,2] **Wendy E. Mackay**[2,1] **Michel Beaudouin-Lafon**[1,2]

[1]LRI, Université Paris-Sud, CNRS,
Université Paris-Saclay
F-91405 Orsay, France

[2]Inria,
Université Paris-Saclay
F-91405 Orsay, France

{mciolfi, maudet, mbl, mackay}@lri.fr

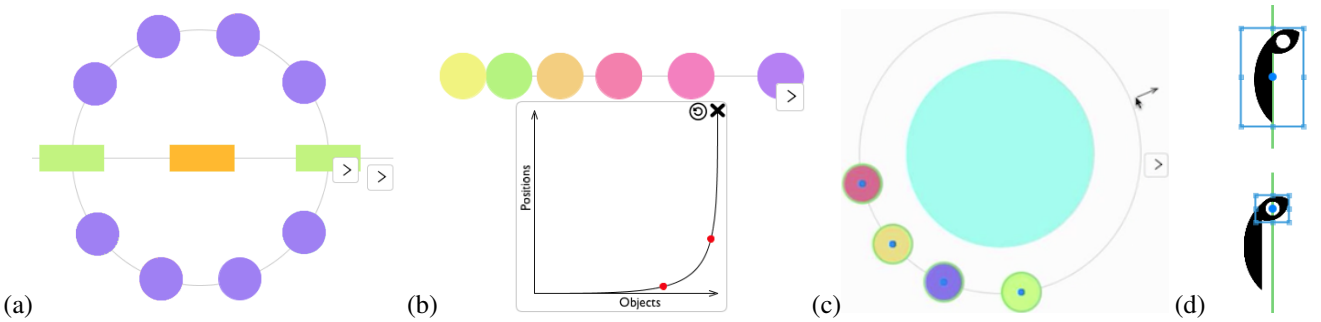

Figure 1: *StickyLines* reify alignment and distribution into first-class graphical objects that users can manipulate directly. (a) Circular and horizontal alignments. (b) Non-linear distribution. (c) Ghost guideline. (d) Tweaking an object's bounding box.

ABSTRACT

Aligning and distributing graphical objects is a common, but cumbersome task. In a preliminary study (six graphic designers, six non-designers), we identified three key problems with current tools: lack of persistence, unpredictability of results, and inability to 'tweak' the layout. We created *StickyLines*, a tool that treats guidelines as first-class objects: Users can create precise, predictable and persistent interactive alignment and distribution relationships, and 'tweaked' positions can be maintained for subsequent interactions. We ran a [2x2] within-participant experiment to compare *StickyLines* with standard commands, with two levels of layout difficulty. *StickyLines* performed 40% faster and required 49% fewer actions than traditional alignment and distribution commands for complex layouts. In study three, six professional designers quickly adopted *StickyLines* and identified novel uses, including creating complex compound guidelines and using them for both spatial and semantic grouping.

Author Keywords

Magnetic guidelines; Snapping; Alignment; Distribution; Tweaking; Graphical authoring; Appropriation.

ACM Classification Keywords

H.5.2 [Information Interfaces and Presentation]: User Interfaces - Graphical user interfaces

INTRODUCTION

Authoring graphical documents, such as posters, slide presentations and diagrams, requires precise positioning of graphical objects to achieve aesthetically pleasing results. Current systems typically include commands that align or distribute selected objects horizontally or vertically. For example, a horizontal *'align-center'* command uses the geometric center of each object to move the selected objects vertically. Similarly, a vertical *'distribute'* command either positions the objects evenly along the vertical axis, using their centers as reference points, or spaces them evenly, using their tops and bottoms to distribute the available space between objects.

Previous studies highlighted the problem of positioning objects. For example, Janacek et al. [18] reported that an expert Colored Petri Net designer spent over 60% of his time in a basic design task performing tedious and repetitive operations to reposition graphical objects. Mackay et al. [20] reported that "*expert users seriously underestimate how much time they spend on minor manipulations of the tool, especially those involving layout*". These designers estimated that they spent approximately 5% of their time on graphical repositioning, but video records showed that they actually spent closer to 30%.

Graphical authoring tools have evolved in the subsequent 15 years. However, have these tools addressed the cumbersome nature of graphical alignment and distribution? Or do users of graphical editing tools still struggle and waste time repositioning objects?

M. Ciolfi Felice, N. Maudet, W. E. Mackay, M. Beaudouin-Lafon.
Beyond Snapping: Persistent, Tweakable Alignment and Distribution with StickyLines.
In UIST '16: Proceedings of the 29th annual ACM symposium on User interface software and technology, ACM, October 2016.
Authors Version.

After first reviewing the related literature, we describe the results of a preliminary study of 12 regular users of graphical editing tools to assess how today's users deal with alignment and distribution. We then describe the design and implementation of *StickyLines*, which use the principles of Instrumental Interaction [2, 5] to *reify* spatial relationships into first-class objects. *StickyLines* extend *magnetic guidelines* [4], reifying not only alignment but also distribution into persistent graphical objects that users can directly manipulate. *StickyLines* also support *tweaking* the positions and bounding boxes of graphical objects, capturing ad-hoc changes made by the user and making them persistent.

We conducted two studies to evaluate *StickyLines*: a [2x2] within-participant experiment that compares the basic capabilities of *StickyLines* with traditional alignment and distribution commands for easy and difficult tasks, and a structured observation study of how designers use and appropriate the more advanced features of *StickyLines*. We conclude with a discussion of the results and directions for future research.

RELATED WORK

Within the extensive previous work on graphical editing, we focus on alignment and distribution, starting with early snapping techniques. Most commercial systems use imperative, command-based approaches, although the research literature has also focused on constraint-based declarative approaches. Finally, more recent work has explored explicit types of reification, such as rulers and guidelines.

Snapping Techniques

As early as 1964, *Sketchpad* [27] featured *gravity fields* to snap the cursor to nearby objects. *Snap-dragging* [6], based on a ruler and compass metaphor, creates transient 'alignment objects' (points, circles and lines) inferred from the elements in the document. Since then, snap-dragging has been extended by manipulating the motor space [1] or the user's velocity profile [11], and new techniques have been introduced, such as keeping objects aligned across slides [10].

Command-based Techniques

Almost all current commercial applications for graphical authoring, such as Adobe Illustrator and InDesign or Microsoft PowerPoint, feature menu-based commands for alignment and distribution. A recent technique, *GACA* [33], can align and distribute objects in 2D if they are roughly aligned, in a single operation. The system infers relationships in the selected set, without having to work with 1D subgroups. However, command-based techniques do not make relationships persistent and their results can be hard to predict by users.

Constraint-based Techniques

Constraints encode relationships among objects in a layout. For example, "The space between these two rectangles must always be at least 5cm", or "Buttons must be left-aligned". Sketchpad [27] was the first interactive tool to integrate constraints and direct manipulation. In most constraint-based approaches, e.g., *Juno* [23], *IDEAL* [28] and *Dunnart* [9], users declare constraints and the system computes a layout that satisfies them. Some systems focus instead on constraint inference, such as *Chimera* [19], *Pegasus* [17] and *Penguins* [7]. For example, *DesignScape* [24] automates *"the tedious parts of design"*, including alignment, by making layout suggestions based on user-defined and system-inferred constraints. Other systems, such as Xu et al.'s beautifier [32] allow users to interact with the inferred constraints.

Geometric constraints, particularly over- and under-constrained configurations, can be difficult to solve and the results can be difficult to anticipate. Wybrow et al. [30] compared *one-way* and *multi-way constraints* for diagram editing. One-way constraints are easy to understand but are limited. Multi-way constraints overcome these limitations, but make the system much more complex. The authors found that alignment and distribution would be more usable if they provided *"truly persistent relationships"*, which are only possible with multi-way constraints.

Combining Snapping and Constraints

Briar [15] combines *snap-dragging* with constraints [16]. Constraints are specified through augmented snapping, which takes the snapping location as an extra parameter to infer constraints. When snapping an object, the system reveals the new possible relationships to the user, who must choose among or reject them. However, the user cannot manipulate the constraints directly, only the objects, and distribution is not supported. *GLIDE* [26] explores a similar approach, representing constraints with 'indicator' objects. In *HyperSnapping* [22], snapping objects creates constraints represented by square 'anchors' at the snapping points, and the snapped objects are treated as a group. Distribution is supported, and the relationships are visible but they cannot be manipulated directly and they are not persistent: the constraints are cleared when clicking outside the group.

Reification of Alignment and Distribution

Several approaches have explicitly reified the concept of alignment into interactive objects. Raisamo's *alignment stick* [25] uses a physical ruler metaphor to push objects. *Lineogrammer* [34], a pen-input system for diagram editing, extends the alignment stick with a 'grabby' ruler that collects objects when passing over them and supports distribution. However, while the stick reifies the action of aligning, the relationships themselves are neither directly manipulable nor persistent. In *Rock & Rails* [29], specific hand gestures represent constraints and help users align objects on a multitouch tabletop. In *Object-oriented drawing* [31], users can create persistent alignments by linking the positions of graphical objects via *attribute objects*. *Magnetic guidelines* [4] reify alignments into persistent graphical objects that users can directly manipulate: objects can be attached and detached from a guideline, and moving a guideline moves the objects attached to it. *Neat* [12] adapts *magnetic guidelines* to a tabletop surface, and Beaudouin-Lafon introduced a version that supports distribution [3].

In summary, many approaches have addressed alignment and distribution, but very few make them visible and directly manipulable. Our approach builds on *magnetic guidelines*. In order to better understand users' needs, we started by conducting an observational study of expert and non-expert users.

STUDY 1: OBSERVATION

We conducted critical object interviews [21] to better understand how regular users of advanced graphical editing tools deal with alignment and distribution. We interviewed twelve regular users (ages 24-38; four women) of Adobe Illustrator, Adobe Photoshop, Sketch, Inkscape, Gimp, Corel Draw, Microsoft PowerPoint and Prezi.

Half the participants (6/12) were professional designers (UX, product, and web designers); the other half included a software developer, a design student, a biologist, a political scientist, a geologist, and a computer scientist. We interviewed them in their offices or homes and invited them to show us recent projects in which they had to lay out graphical objects. We asked them to recall specific problems, focusing on *breakdowns* (interactions that led to unexpected or incorrect results), *user innovations*, and *appropriations* (the personal strategies they used, especially when dealing with breakdowns). We also encouraged them to show us these problematic tasks, and observed their reactions to mismatches between their expectations and the system's behaviour.

We used a grounded theory approach [14] to classify the collected stories. Grounded theory offers a systematic approach for analysing qualitative data: instead of testing hypotheses, the analysis is guided by a series of open-ended questions and rigorous strategies for guiding the data collection. The goal is to provide a rich description of the phenomena that emerge. We identified specific problems in each story and cross-checked them with the other stories. This provided us with a list of the key problems these users face when aligning and distributing graphical objects, as well as an indication of how common they are.

Results

Many participants (8/12) often aligned and distributed objects 'manually': They first used the mouse to roughly position the object, and then used the arrow keys to visually fine-tune it. Participants view alignment and distribution commands as 'automatic' operations, and treat everything else as 'manual' operations, including using rulers (8/12), making visual comparisons within a zoomed-in area (7/12), and typing in coordinates (2/12). All the designers and one developer (P12) make extensive use of the keyboard to align and distribute objects, not only because it is faster, but also because *"there are too many options and menus"* (P3, UX designer) that clutter their screens and make them *'lose focus'* (P2, web developer/designer). Three non-designers mentioned that aligning and distributing are highly time consuming.

Not surprisingly, designers are more concerned with accuracy than non-designers; half (3/6) still use online tutorials when faced with complex tasks. Five participants felt their alignment and distribution strategies could be improved: *"There must be a better way to do it, but this is my solution."* (P6).

Sometimes, when participants encounter an obstacle, they 'cheat' by using a tool in an unorthodox way. For example, P2 (web developer/designer) needed to ensure equal spacing among a series of objects: *"I do not understand the distribution commands, so what I did was to cheat. I put one object next to the right side of the first one, I selected it and then pressed shift and the right arrow. I counted how many times I pressed the arrow, this gave me a kind of procedural measure of the space between the objects that I memorised and then repeated for the rest"*. P10 (biologist) used a similar procedure, because *"it is safe"*.

Many participants want to make the relationships among objects explicit, so they can be visualised, manipulated, and captured for later editing. For example, half the participants reused a previous alignment or distribution by duplicating the objects and replacing them with new ones, even though P9 (computer scientist) considered this to be *'cheating'*. P7 (political scientist) and P10 (biologist) wanted to know the distance between two graphical objects: *"The grid is not enough, I cannot count the squares."* (P7). P8 (geologist) needed to add tags to several pictures at the same position relative to their frames: *"I wish I had a way to declare this to the program."*. P9 (computer scientist) wanted equal spacing among items and created an *invisible spacer* – a transparent rectangle with the same height as the space he wanted to duplicate. P11 (design student) created her own spacer by *"cutting the distance between two objects and pasting it between the rest of them."*.

We identified three key issues that users face when positioning objects using current graphical authoring tools:

- Lack of *persistence*: The system does not keep objects aligned or distributed, forcing users to realign or redistribute them after every minor change.
- Lack of *control*: Users often cannot predict the results of their commands. Users also lack tools for making and preserving minor corrections or 'tweaks'.
- Lack of *generality*: Users are limited to horizontal and vertical layouts when aligning and distributing objects.

Persistence

In command-based tools, applying an alignment or distribution command moves the objects but leaves no concrete trace of its use. Any change to one of the objects will likely require the user to reapply the command. This lack of persistence leads to the repetition of actions and hinders the reuse of previous results. For example, P5 (web developer/designer) aligned two objects vertically: *"I wanted to move one to the right. I wish I could do it only in the horizontal axis, instead of being worried about introducing an offset in the vertical one. Some constraints are obvious to me but they are not captured by the tool, so it gives me more freedom than I need, and I have to realign."*.

The lack of persistence is closely related to the need to support *repetitive* not just *one-time* tasks. Optimising repetitive tasks requires some planning, such as creating auxiliary structures or guides, which is not worth it for most one-time tasks. P9 (computer scientist) explained that *"you need to have an idea of how the objects should look, and only then align with the commands, not the opposite; so you either plan everything in advance, or you reapply everything you did."*. P3 (UX designer) explained that *"for a one-time thing I do the job manually, but for a frequent task I find a tutorial to learn*

how to solve it with tools.". Most participants (8/12) appreciate the *automatic guidelines* that appear in some graphical editors, even though they are not persistent: As the user drags an a object, when a guideline appears, they can snap the object to it simply by releasing the mouse. Two of the other participants were unaware of this feature, one did not have it in his editor, and one had disabled it.

Control

The icons used to depict alignment and distribution commands appear intuitive, but users still have difficulty predicting the results. P1 (designer) was trying to distribute objects and the outcome was not what he expected: *"It is not clear what will be the effect of the command, even if you have some experience with the tool. It is normal to have to undo and retry, sometimes it does not do what you want. See? This does not make sense to me. I am not even sure if I chose the right command."*. P9 (computer scientist) wondered: *"I am aligning with respect to what? Does the selection order matter?"*. P10 (biologist) had the same problem with distribution: *"What is the reference? Is it the width of the page?"*. P9 (computer scientist), after successfully aligning a group of objects inside containers, added: *"Now I was lucky, sometimes I have to undo and repeat the action, because it moves the element or the box. I have to be always alert, and do it in a precise mechanical way, always thinking of making the selection in the correct order."*.

Current command-based systems do not reveal how their algorithms work. Few highlight the alignment's *pivot* (the object used as a reference to align other objects to it) or the object's *anchor* (the reference point within an object used for alignment – usually the object's center or a side), and even fewer let the user choose them. Users cannot pre-determine if or how the selection order will affect the output. Half the participants did not feel in control and were frustrated by the commands, which they described as *'awkward'* (P12) and *'too automatic'* (P4, P5).

P5 described annoying limitations of the tool: *"There is a problem with hierarchy in layers and groups. Sometimes I cannot directly relate an object to one in another group, because they do not see each other; I have to ungroup and regroup so that the tool lets me align them."*. These breakdowns caused P10 (biologist) to completely lose faith in commands: *"Align vertically always makes a disaster. I do not trust it, so I do not trust align centers either."*. P12 (developer) also felt the loss of control: *"I have more trust in moving things manually because I find it more practical, I can put them exactly where I want."*.

Some participants came up with clever tricks. P5 (web developer/designer) puts his icons and labels inside transparent square containers that are larger than the icons, which he keeps aligned: *"The white space between an object and its square generates the illusion of space between two icons, but in reality it is a fake space, the containers are next to each other, so it is easy for me to locate them in regular positions. I have 100% control over what happens."*. P9 (computer scientist) described a similar strategy: *"Look how I cheat. I create a fictitious box with a certain alpha, but not transparent, with a distinctive colour, very different from the background so it highlights and I remember it is not a real object. Then I center each icon in its box, I group each pair, and I align the boxes."*.

Alignment and distribution commands use the geometric center of objects, but sometimes this does not match the object's visual center. Seven participants had recently used commands to align what they referred to as *'irregular'* or *'weird'* shapes, including icons, logos and text within a graphic design. All were forced to fine-tune the result to make it aesthetically pleasing. We call such edits *tweaks*. For example, P3 (UX designer), P5 and P6 (web developers/designers) switched to a grid view and manually arranged each object's position. To our knowledge, current tools completely ignore such tasks, so users must perform them manually after each use of an alignment or distribution command, therefore increasing the need for repetitive actions, preventing output reuse and increasing the likelihood of errors.

Generality

Sometimes users want to align objects along a diagonal, or shapes other than a straight line. They may also want objects, such as the arrows in a diagram, to remain parallel in spite of future edits. However, most current tools are limited to horizontal and vertical alignment and distribution. The exceptions include tools that allow text to be wrapped onto custom shapes, or distribute objects equally relative to their centers or sides.

P12 (developer) had to align text and images at different angles. Due to the lack of tool support he had to check visually if they looked right. P5 (web developer/designer) was working on a wheel-shaped menu, with icons in the center of each slice. He had to create an *'icons guideline'*, a layer with a grey circle that served as a visual guide to place the icons. This *guideline* can be seen as a reification of the relationship among the icons in the circular menu, i.e. a concrete object he could interact with. P3 (UX designer), and P11 (design student) used similar strategies.

Summary: Reifying Alignment and Distribution

Current tools lack *explicit* representations of alignment and distribution. We argue that both should be represented as first-class objects that users can manipulate directly. These new objects would then be persistent, and have their own settings and properties, like other graphical objects. Finally, these alignment and distribution objects could be made more flexible and powerful, for example to support circular alignments or non-linear distributions.

The process by which an abstract concept, such as alignment, is turned into a first-class object is called *reification* [5]. We propose to reify alignment and distribution into a new category of objects called *guidelines*. Unlike current rulers that help users adjust objects by eye, these guidelines are active relative to the objects, and can be easily manipulated and maintained. *Magnetic guidelines* [4] introduced this approach in the context of a Colored Petri Net design tool. However, the capabilities were limited and not evaluated formally, and were not integrated into a general purpose tool.

STICKYLINES

We created *StickyLines*, a graphical editor with persistent, tweakable guidelines in order to address the users' requirements for persistence, control, and generality. *StickyLines* features a canvas and a tool palette. Users can create standard geometric shapes, import images, and move and resize them by direct manipulation.

Guidelines

Users create guidelines using five tools in the palette. For horizontal, vertical, circular and parallel guidelines, they simply click in the canvas and the guideline appears. A *parallel guideline* is a line at any angle that keeps objects parallel to each other, perpendicular to the line. To create a *ghost guideline*, users must click an object. The guideline takes the object's shape and the user can adjust its offset (Fig. 1c).

Dragging an object close to a guideline highlights the *snap point* (center or side) that would be used for alignment if it were dropped. Dropping the object attaches it to the guideline at that snap point. Dragging an object away from a guideline it is attached to detaches it.

Guidelines can be manipulated like regular objects: they can be resized, moved, and deleted. Moving a guideline also moves the objects attached to it. Deleting a guideline leaves the attached objects at their current position. A guideline can be reshaped into another form and the positions of the attached objects adapt to the new form. Each guideline has a button that opens a palette with additional tools, such as distribution (see below) and reshaping.

Multiple Relationships Per Object

An object can be attached to several guidelines by moving it close to them. When the cursor hovers over an object, the guidelines to which it is attached are highlighted. When the user creates or releases a guideline near an object, the object snaps to it unless this breaks an existing relationship, i.e. unless the object was already attached to another guideline and snapping it to the new one would change the object's position. This enhances predictability and reduces the chances of changing previous alignments and distributions by accident.

When the user moves a guideline, the system tries to preserve existing relationships, but the guideline being moved takes priority in case of conflicts. For example, moving the horizontal guideline upward in Fig. 1a will detach the two rectangles from the circle once the guidelines no longer intercept.

For added control, users can open a guideline's palette and give it high priority, so that it is not overriden. Users can also set the priority of individual snap points by double clicking the snap point or by clicking on the star that appears when selecting it. In order to resolve conflicts when moving a guideline, the system uses the most recent priority of each type of guideline the object is attached to and of each snap point of the object. For example, if the object has both its left and right sides attached to vertical guidelines with high priority and the user moves one of them, the guideline whose priority was set most recently will prevail. High priority relationships are displayed in orange, to help the user anticipate the system's behaviour. In addition, users can set a guideline as 'exclusive' in the guideline's palette, so that its attached objects will ignore all other guidelines. An outline of the layout algorithm when moving a guideline can be found in the appendix.

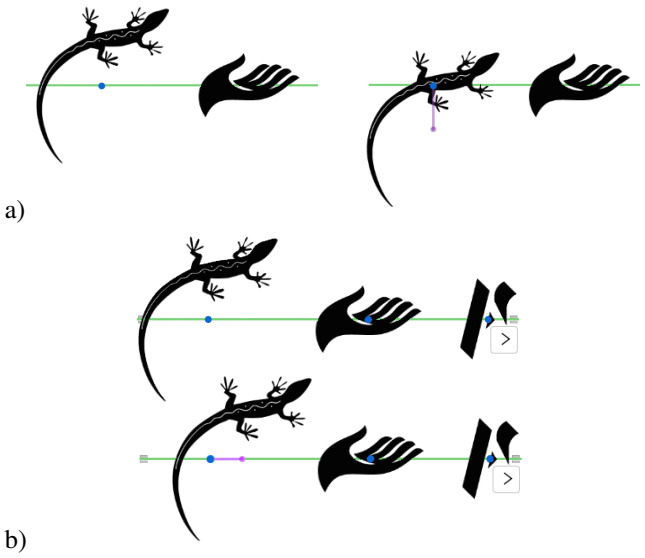

Figure 2: Tweaking the reference point (in blue) to (a) align or (b) distribute according to the visual center. The tweak is displayed in purple and can be manipulated directly.

Automatic Creation and Deletion of Guidelines

When 'auto-create' is active in the tool palette, *StickyLines* detects potential horizontal, vertical and parallel alignments while objects are being moved, and displays dotted guidelines to provide transient feedforward. Dropping the object while this feedforward is visible automatically creates a guideline.

When 'auto-cleanup' is active in the tool palette, guidelines that become empty are automatically deleted, to avoid cluttering the screen. Users can also hide all guidelines from the tool palette. Hidden guidelines remain active: hovering the cursor over an object attached to a hidden guideline highlights the other objects on that guideline; Objects can be attached to a hidden guideline by moving them close to it.

Distribution

Users can select a distribution type from the guideline's palette to distribute objects along the full length of a linear guideline or the perimeter of a closed guideline. Options include equal spacing among objects or equal distances among reference points, e.g., left, center or right for horizontal distribution. Reference points highlight when the cursor hovers over the guideline or its attached objects.

When distribution is active, the layout is recomputed whenever objects are added or removed from the guideline. Users can also directly manipulate a curve that represents the mapping between each object and its position along the guideline, making it possible to create non-linear distributions (Fig. 1b).

Tweaking the Reference Point

Users can *tweak* the placement of objects attached to guidelines in order to, e.g., correct an alignment when the visual

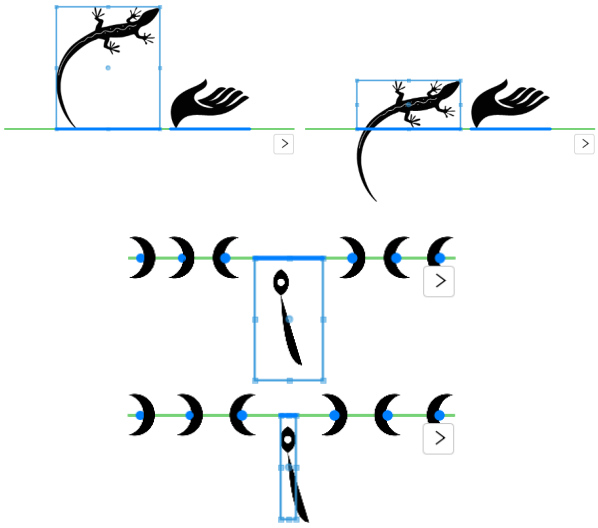

Figure 3: Tweaking the bounding box to base (a) alignment or (b) distribution on the visual extent.

center of an object is not its geometric center (Fig. 2a), or to adjust the result of a distribution without modifying the position of other objects on the guideline (Fig. 2b).

Moving the arrow keys to 'nudge' an object visually repositions it, even though the object remains logically attached to the guideline. This offset, called a *tweak*, is recorded and displayed. The tweak is persistent: moving the guideline preserves the offset. Tweaks belong to the objects so that if an object is detached from a guideline, its tweak will be reused when attaching the object to another guideline.

Tweaks reify the action of adjusting an object's position, which is often needed when fine-tuning a layout. They are first-class objects that can be edited, copied onto other objects, and deleted. Tweaks are normally shown only when interacting with their parent object, but a tool in the palette lets users display all the tweaks.

Tweaking the Bounding Box

Users can also tweak the bounding box of an object in order to, e.g., finely control its placement on a guideline when it is attached by one of its sides (Fig. 3). Hovering the cursor over an object displays its bounding box. The geometric bounding box is the default, but users can resize and move it through direct manipulation, without affecting the object itself. When hovering the cursor over a bounding box, its associated object is highlighted if the two do not overlap.

Moving and resizing an object moves and resizes its bounding box. Bounding boxes can be copied onto other objects, replacing their current one. In the same way as tweaking the reference point reifies adjustments to the object's position, tweaking the bounding box reifies adjustments to its extent.

Summary

StickyLines explicitly addresses the requirements identified in the first study:

- Guidelines and tweaks are *persistent*;
- Guidelines are visible and directly manipulable, enhancing user *control* over layout;
- Guidelines offer more *general* support for alignment and distribution operations and support tweaking to mediate between automated layouts and ad-hoc modifications.

Our early experience with *StickyLines* convinced us that they facilitate the creation of complex layouts and encourage creativity. For example, a designer may first organise a graphical structure by creating some guidelines, populate them with objects and 'play' with the layout by moving the guidelines and tweaking the objects. To support this claim we conducted two studies: a controlled experiment to compare *StickyLines* to standard command-based alignment and distribution, and a structured observation of designers using *StickyLines* for a set of realistic tasks.

STUDY 2: STICKYLINES VS. COMMANDS

A key feature of *StickyLines* is to use guidelines to support alignment and distribution instead of the menu commands available in traditional tools. We therefore decided to run a controlled experiment to compare traditional commands with guidelines. Since many of the novel features of *StickyLines*, such as tweaking, have no equivalent in standard tools, we chose to compare only horizontal and vertical alignment and equal distribution of space and reference points.

Our hypothesis is that guidelines are more efficient than commands when creating complex layouts, i.e. they are faster and require fewer operations. We expect the differences to be larger for more complex layouts, when multiple adjustments are required. Indeed, if a given layout can be obtained by using an alignment command only once, a guideline provides little advantage. It may even take more time to create the guideline and attach the objects than to invoke a traditional align command. By contrast, adjusting the horizontal position of a vertical alignment can be done simply by dragging the guideline, while a command-based system requires grouping or selecting the objects and moving them. The experiment attempts to determine whether or not this benefit is sufficient to provide a significant advantage to guidelines in practice.

Experimental Design

We use a [2x2] within-participant design with two primary factors: TECHNIQUE (*Command* or *StickyLine*) and DIFFICULTY (*Easy* or *Hard*). The two levels of difficulty (Fig. 4) are operationalised by the dependencies among the alignments and distributions to be created in a target layout. More precisely, we define a layout's optimal solution as the minimum number of actions (such as applying an alignment command, or attaching an object to a guideline) required to complete it. To achieve the optimal solution, *hard* tasks require performing the actions in a certain order, while *easy* tasks do not impose a particular order. For example, the layout in Fig. 4b requires distributing the circles vertically, as indicated by the blue line, and only then aligning them with the rectangles on the right. By contrast, the alignments marked in red (Fig. 4a) can be performed in any order.

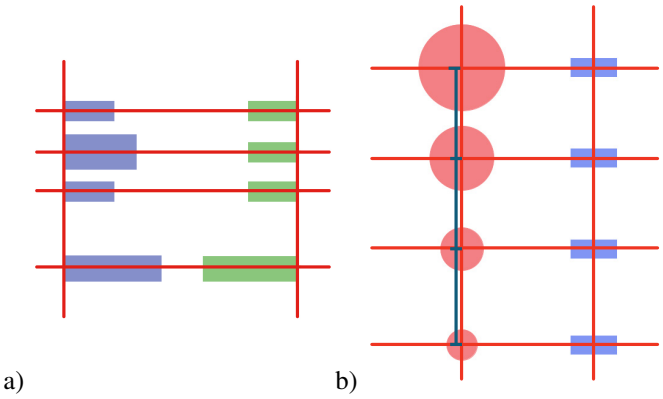

Figure 4: Trial examples: (a) Easy layout; (b) Hard layout

Participants perform three tasks in each of the four conditions, i.e. a total of twelve tasks. In order to avoid learning effects, we use two sets (*A* and *B*) of six layouts, each with three *Easy* and three *Hard* layouts. For each level of difficulty, layouts in sets *A* and *B* require the same number of actions. Sets *A* and *B* are counterbalanced by TECHNIQUE across participants. This ensures that participants are exposed to different layouts for each level of difficulty in each of the two TECHNIQUE conditions and for each repetition.

Participants

We recruited 12 participants (ages 22-34; five women, seven men) with normal or corrected-to-normal vision.

Apparatus

We created a Java application that supports command-based alignment and distribution similar to standard tools, as well as a subset of *StickyLines*' features. The experiment is run on a 13" MacBook Pro running Mac OS 10.11. Participants can use the mouse or trackpad according to their preference.

The *Command* condition includes a tool palette with six alignment commands (three horizontal and three vertical, one for each reference point), eight distribution commands (three horizontal and three vertical, one for each reference point, plus horizontal and vertical equal spacing), and a reset button to start over from the initial position of the objects.

The *StickyLine* condition includes only a palette with tools to create horizontal and vertical guidelines, and the reset button. Guidelines can be resized, moved or deleted, and objects can be attached or detached by direct manipulation. Users can activate or deactivate an equal distribution of space or reference points by choosing it from the guideline's palette. None of the other features of *StickyLines* (feedforward, reshaping, distribution curves, ...) are included.

Procedure

Participants first receive a live scripted demonstration of the tool and practice the two techniques for five minutes. The DIFFICULTY conditions are grouped by TECHNIQUE and both factors are counterbalanced across participants. The order of techniques during practice is also counterbalanced across participants. Half the participants view set *A* for the first TECHNIQUE and set *B* for the second, the other half view layouts in the reverse order. Three replications of each DIFFICULTY×TECHNIQUE condition result in a total of 12 trials per participant. For each trial, participants are given a printout of the target layout showing the alignments and distributions (Fig. 4) and are told to create it as quickly and as accurately as they can. Participants decide when they are done, and then fill out a short questionnaire. The experiment takes about 45 minutes.

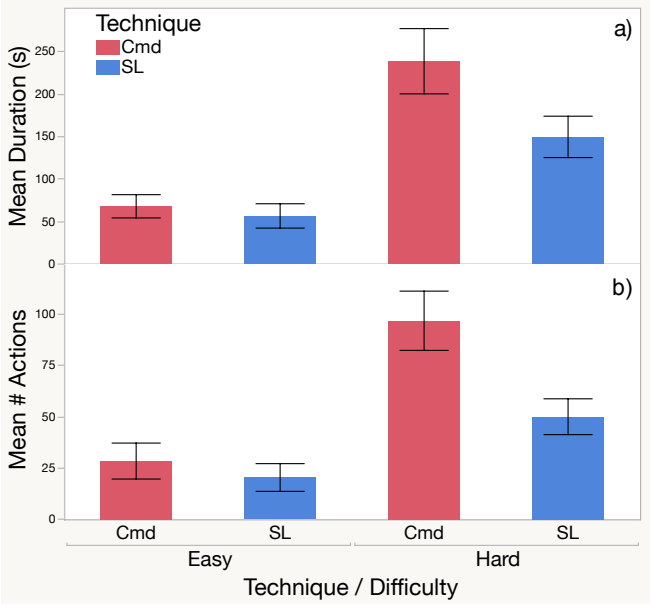

Figure 5: a) Mean duration and b) Mean number of actions by TECHNIQUE × DIFFICULTY, with 95% confidence intervals.

Data Collection

We collected data for $2 \times 2 \times 3 \times 12 = 144$ trials. Measures include the *DURATION* of each trial in seconds and *#ACTIONS*, the number of elementary user actions such as move, align or create a guideline. We also collected a log of low-level mouse and keyboard events and recorded the screen.

Results

Since both *DURATION* and *#ACTIONS* are strictly positive, we use a log transform of both measures in the rest of the analyses. We observe that the transformed dataset exhibits no outliers and is normally distributed. We also observe no significant effect of layout set (*A* or *B*) and no learning effect across repetitions. We let participants use their preferred device: seven used the mouse and five the trackpad. A t-test shows that the input device has no effect on *DURATION* ($t(116) = -0.44, p = 0.66$) nor *#ACTIONS* ($t(120) = -0.59, p = 0.55$).

An ANOVA[1] in the model *DURATION* ~ TECHNIQUE × DIFFICULTY × Rand(PARTICIPANT) shows significant main effects of TECHNIQUE ($F_{1,11} = 16.25$, $p = 0.002$) and DIFFICULTY ($F_{1,11} = 165.02$, $p < 0.0001$), and a significant TECHNIQUE×DIFFICULTY interaction ($F_{1,11} = 6.02$, $p = 0.032$) (Fig. 6a).

[1] All analyses are performed with SAS JMP, using the REML procedure to account for repeated measures.

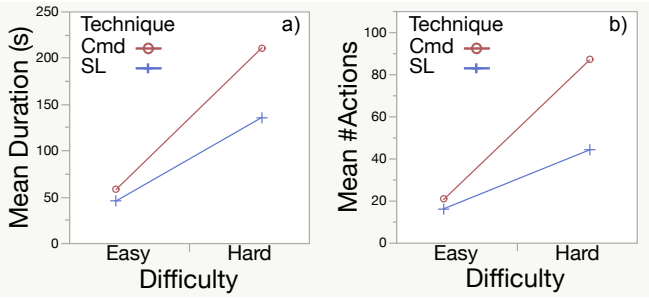

Figure 6: Interaction effects between TECHNIQUE and DIFFICULTY for (a) duration and (b) number of actions.

Before performing post-hoc Tukey HSD tests, we check the normality hypothesis for each group and the homogeneity of variance hypothesis among groups. Goodness-of-fit tests for lognormal distributions are all positive (Kolmogorov's D test, $p = 0.15$). Homogeneity of variance is also verified (Brown-Forsythe, $p = 0.73$). Post-hoc Tukey HSD tests show that *DURATION* is not significantly different between techniques for *Easy* layouts (59 seconds for *Command* vs. 48 for *StickyLine*), but is for *Hard* layouts (253 seconds for *Command* vs. 152 for *StickyLine*), supporting our hypothesis (Fig. 5a).

A similar ANOVA for *#ACTIONS* shows similar results: significant main effects of TECHNIQUE ($F_{1,11} = 25.95$, $p = 0.0003$) and DIFFICULTY ($F_{1,11} = 147.28$, $p < 0.0001$), and a significant TECHNIQUE×DIFFICULTY interaction ($F_{1,11} = 11.70$, $p = 0.0057$) (Fig. 6b). Post-hoc Tukey HSD tests show that *#ACTIONS* is not significantly different between techniques for *Easy* layouts (20.5 actions for *Command* vs. 16.2 for *StickyLine*), but is for *Hard* layouts (87.6 actions for *Command* vs. 44.5 for *StickyLine*), supporting our hypothesis (Fig. 5b). However these results should be further investigated since the normality hypothesis is verified by *Command-Hard* ($p = 0.15$) and *StickyLine-Easy* ($p = 0.13$) but not *StickyLine-Hard* ($p = 0.04$) and *Command-Easy* ($p = 0.01$), and the homogeneity hypothesis is rejected (Brown-Forsythe, $p = 0.04$).

Discussion

The results highlight the effectiveness of *StickyLines*, especially for complex layouts. They reduce the time for creating a difficult layout by approximately 40% (from 253 to 152 seconds) and the number of actions by 49% (87.6 to 44.5). However, the differences are much smaller for simple layouts, and are not significant. Although we expected that guidelines would be faster for complex layouts, we did not know if they would outperform alignment and distribution commands for simple layouts. This is because creating a guideline and then dragging each object to it may take more time, with more actions, than selecting the objects and then the align command. However our results suggest that guidelines are not detrimental for simple layouts.

The event log and screen recordings show that in the *StickyLine* condition, participants use guidelines extensively, allowing them to progressively create the layout. By contrast, in the *Command* condition, they use alignment commands more sparingly, and in fact they sometimes give up and create alignments purely visually. To assess the prevalence of this behavior, we used the event log to count the number of times in which the *last* action applied to an object in the *Command* condition is a move as opposed to an align or distribute command, indicating that the participant assessed the object's final position visually. In 89.5% of the cases, the last action for a given object is an alignment or a distribution. In 4.9% of the cases, it is a move of a single object, i.e. a visual alignment. The remaining cases are ambiguous: 4.2% are a move of a group of objects and 1.4% are a constrained horizontal or vertical move of a single object, both of which may be used to maintain an alignment. By contrast, in the *StickyLine* condition, only two occurrences (0.4%) of final visual alignment of an object were recorded in the log.

In the post-hoc questionnaire, participants ranked guidelines as easier, more enjoyable and faster to use than commands, supporting the quantitative results. However two participants also found them more mentally demanding and one found them more frustrating. This may be due to a higher familiarity with command-based alignment or to some idiosyncrasies of our implementation. It may also be due to the fact that guidelines can require more planning to create the proper structure, while with classical tools users often resort to visual, and therefore approximate, alignment.

In summary, this experiment supports our hypothesis that guidelines are an efficient and powerful alternative to traditional commands. However, since it covers only a subset of *StickyLines* features, we also conducted a structured observation study to assess how professional designers use *StickyLines*' more advanced capabilities in a realistic setting.

STUDY 3: STRUCTURED OBSERVATION

We are interested in how users interact with *StickyLines* when they need to create and tweak complex structures that will be reused. We want to observe how designers use and appropriate the advanced features of *StickyLines* that give them more control over alignment and distribution, and capture their strategies. Since our first study showed that designers have higher accuracy requirements than non-designers, we conducted a structured observation of expert use with six designers. Structured observation [13] is a type of quasi-experiment [8] designed to enhance external validity by combining controlled conditions that facilitate comparisons within and across realistic tasks.

Participants

We recruited six designers (ages 22-30; all women) with normal or corrected-to-normal vision. All have a Bachelor's or a Master's degree in design, with two to seven years of experience. All participants are regular users of Adobe Illustrator and Adobe Photoshop, and some of Adobe InDesign (5/6), Sketch (3/6), Corel Draw (1/6), and Inkscape (1/6).

Apparatus

The version of *StickyLines* used in this study includes most of the features described earlier: horizontal, vertical and circular guidelienes, distribution (of space or reference points), reshaping, tweaking reference points and bounding boxes, hiding/showing guidelines and tweaks. In order to keep the training time to around ten minutes and to avoid overwhelming

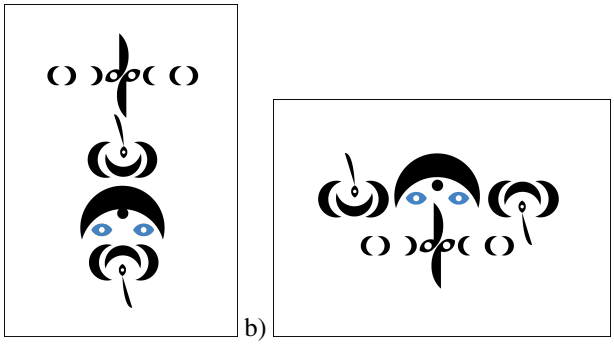

Figure 7: Posters to reproduce: (a) portrait and (b) landscape.

participants with a number of features that are not at the core of the tool, we did not include parallel and ghost guidelines, and disabled feedforward, distribution curves, and automatic guideline removal. The study is run on a 13" MacBook Pro with Mac OS 10.11. Participants can use the mouse or trackpad according to their preference.

Procedure

Each participant receives ten minutes of training with *StickyLines*, followed by two minutes of free practice. The study includes three tasks, and uses a think-aloud protocol. At the beginning of the first task, participants are given two printed posters and asked to reproduce them using the predefined objects displayed on the screen (Fig. 7). They are told that each task builds upon the results of the previous one – including guidelines, tweaks and bounding boxes – unless they prefer to reset the layout. The posters include ambiguous alignment and distribution relationships, as well as 'irregular' shapes, to encourage diversity in the solutions. In task one, participants reproduce the first poster. In task two, they can reuse the result of task one to generate the second poster, which is a similar layout, but with a different page orientation. Half of the users convert from portrait to landscape, the other half does the opposite. The first two tasks are not timed. In task three, participants have ten minutes to continue the series by designing two more posters that they would present to a client. The three tasks take approximately 45', after which participants complete a post-hoc questionnaire.

Data Collection

We recorded the screen and the audio and took notes. We logged the interaction of the participants with the tool and we collected the answers to the post-questionnaire.

Results and Discussion

StickyLines as Instruments to Structure and Modify Layouts

All participants relied extensively on *StickyLines* to construct their layout in task one and to adapt it later for tasks two and three. In task one, most participants (5/6) used the same strategy i.e. to *"first create a guideline for the main structure"* (P3) and later create secondary guidelines. For example P1 created a vertical *'base mark'*, rougly positioned all the objects, and only then added the other guidelines. By contrast, P2 first created all the guidelines she thought she would need before manipulating any object. She then 'collected' objects by releasing a guideline close to them, in sequence.

In task two, all participants reused existing guidelines, and P1 and P5 stated that they were useful. Participants also used *StickyLines* to verify alignment: P4 created a vertical guideline close to two objects that were already positioned in horizontal alignments, to check whether or not they were also aligned vertically.

Tweaks to Adjust Alignments and Distributions

As expected, most participants (5/6) tweaked object positions: all tweaked alignment and one (P5) also tweaked distribution. P1 based her strategy almost exclusively on tweaking reference points, barely using bounding boxes. Not surprisingly, participants created more tweaks in task one than in task two, indicating that they reused their previous tweaks. When converting the poster, P5 appreciated the persistence of tweaks: *"It helps that the tweaks are still there."*.

Among the participants who created tweaks, some (3/5) edited them more often in task two than in task one. Only one participant copied tweaks (P2), in task three, which was more open-ended and exploratory. However, she pasted these two tweaks 16 times, a strong example of reuse. The low use of copying is probably due to the fact that the layouts required mirroring a tweak after pasting it, which is not currently supported by *StickyLines*, so participants decided to create new tweaks instead. More than half (4/6) expressed the need for a *'mirroring'* feature.

Bounding Boxes to Adjust Alignments and Distributions

All participants also tweaked bounding boxes. P3 relied on this feature extensively and did not tweak reference points. In task two, participants reused tweaked bounding boxes more often than tweaked reference points: A majority of participants (4/6) copied bounding boxes and pasted them onto several objects, most during task one (only one in task two).

Participants were not only interested in modifying the perceived borders of objects, but also their perceived centers. P4 modified the bounding box to position its center at the visual center of the object. She then used this new point to attach the object to a guideline.

Tweaks as Grouping Mechanisms

We observed that most participants (5/6) perceived guidelines not only as an alignment and distribution feature, but also as *'groups'* or *'structures'*. They appropriated tweaks to attach objects to a guideline even if they were far away from it, in order to semantically group objects together. (We refer to this as 'super tweaking'.) For example, P3 explicitly used a guideline as a grouping mechanism rather than as an alignment feature. She stated: *"I think of these four objects as a group but this one is not on the guideline"*, so she attached the object temporarily to be able to move the whole group by dragging the guideline, and also to remember that they belonged together, since she was planning to come back later to that part of the layout.

Most participants (5/6) resized guidelines to avoid overlapping other objects and manipulated each one as a small, compact group that they could easily move around. In task 3, half the participants (3/6) moved the guidelines out of the frame

to build the new poster based on their current structures. Half the participants (3/6) hid the guidelines at the end of each task, to compare their work with the printout.

StickyLines as First-class Objects

Participants used guidelines extensively during the three tasks. For the second alternative poster in task three, all participants manipulated the position and type of the guidelines more than the objects themselves, supporting the idea that participants perceive *StickyLines* as first-class objects.

In fact, participants asked for even greater levels of interaction with *StickyLines*. For example, one participant wanted to *"capture the distance between two guidelines in order to reuse it"* (P1), and two participants said that they would like to align and distribute guidelines as if they were regular objects (P1, P4). P4 wanted to cut a line in two parts, since her *"two groups are on the same line"* (P4). Half the participants (3/6) also wanted to be able to merge guidelines. Some participants wanted to know if an object is at the center of a guideline (2/6), to move guidelines precisely with the arrow keys (2/6), to move multiple guidelines at once (5/6), to copy a guideline to reuse its length (1/6), to snap the center of a bounding box to the center of the object (1/6), to reveal all the bounding boxes in the layout (1/6), and to draw the guidelines themselves to define their initial length (1/6). These suggestions demonstrate the power of using guidelines to reify layout relationships, and merit future exploration.

At the end of the study, we asked participants to compare *StickyLines* with their usual tool for creating posters. All participants stated that *StickyLines* was as or more enjoyable than their usual tool, over half (4/6) that it was more powerful and more flexible, and half (3/6) also found it easier to use. P4 stated: *"As with any tool that I have learned, I need time to get my bearings and acquire habits."*. One participant ranked *StickyLines* as less precise than her usual tool because she would move an object's bounding box accidentally when trying to move the object. (The study's version of *StickyLines* did not support zooming.) Three participants found *StickyLines* more mentally demanding. P5 clarified: *"For now, StickyLines is more demanding, but it is also because I am learning it, but [it is] much more powerful and interesting"*.

In summary, study three demonstrates that trained designers can quickly learn to use *StickyLines* and adapt their work practices to take advantage of guidelines and tweaking. It supports the findings from study one about the value of supporting persistence, control and generality to extend the power of tools for graphical layout. Study three also reveals examples of spontaneous appropriation, such as 'super tweaking' an object's position in order to attach it to a distant guideline.

CONCLUSION AND FUTURE WORK

Aligning and distributing graphical objects is a common, but cumbersome task. Our first study shows that users struggle with the alignment and distribution commands provided by current tools and would like the relationships among objects to be persistent, easier to control and more general. We introduce *StickyLines*, a tool that combines persistent guidelines with fine-grained control over alignment and distribution, as well as more general capabilities, such as tweaking an object's position and bounding box. We conducted a controlled experiment showing that, for complex layouts, *StickyLines* are up to 40% faster than standard commands and reduce the number of actions by up to 49%. We also conducted a structured observation study that demonstrates how professional designers can quickly adapt to and appropriate *StickyLines*.

StickyLines relies on the reification of alignments and adjustments, turning them into first-class objects that users not only learn to use efficiently, but also want to push further. This work should encourage designers of graphical tools to incorporate guidelines and tweaking into their tools, and to make both guidelines and tweaks first-class objects. In future work, we plan to expand *StickyLines* to, for example, support symmetrical structures and the alignment and distribution of guidelines themselves. We will continue to investigate how applying the principles of reification, polymorphism and reuse from Instrumental Interaction [5] can significantly improve how designers and other users create and manipulate graphical structures.

ACKNOWLEDGMENTS

This research was supported by ERC grant n° 321135 CREATIV: Creating Co-Adaptive Human-Computer Partnerships. Our thanks to the participants for their creative insights and suggestions about how to use *StickyLines*, and to Inria's ExSitu research lab for fruitful discussions throughout the project.

REFERENCES

1. Baudisch, P. The Cage: Efficient construction in 3D using a cubic adaptive grid. In *Proceedings of the 9th Annual ACM Symposium on User Interface Software and Technology*, UIST '96, ACM (1996), 171–172.

2. Beaudouin-Lafon, M. Instrumental interaction: An interaction model for designing post-WIMP user interfaces. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '00, ACM (2000), 446–453.

3. Beaudouin-Lafon, M. Designing interaction, not interfaces. In *Proceedings of the Working Conference on Advanced Visual Interfaces*, AVI '04, ACM (2004), 15–22.

4. Beaudouin-Lafon, M., and Lassen, H. M. The architecture and implementation of CPN2000, a post-WIMP graphical application. In *Proceedings of the 13th Annual ACM Symposium on User Interface Software and Technology*, UIST '00, ACM (2000), 181–190.

5. Beaudouin-Lafon, M., and Mackay, W. E. Reification, polymorphism and reuse: Three principles for designing visual interfaces. In *Proceedings of the Working Conference on Advanced Visual Interfaces*, AVI '00, ACM (2000), 102–109.

6. Bier, E. A., and Stone, M. C. Snap-dragging. *SIGGRAPH Comput. Graph. 20*, 4 (Aug. 1986), 233–240.

7. Chok, S. S., and Marriott, K. Automatic construction of intelligent diagram editors. In *Proceedings of the 11th Annual ACM Symposium on User Interface Software and Technology*, UIST '98, ACM (1998), 185–194.

8. Cook, T., and Campbell, D. *Quasi-experimentation: design & analysis issues for field settings*. Houghton Mifflin, 1979.

9. Dwyer, T., Marriott, K., and Wybrow, M. Dunnart: A constraint-based network diagram authoring tool. In *Graph Drawing*, I. Tollis and M. Patrignani, Eds., vol. 5417 of *Lecture Notes in Computer Science*. Springer Berlin Heidelberg, 2009, 420–431.

10. Edge, D., Gulwani, S., Milic-Frayling, N., Raza, M., Adhitya Saputra, R., Wang, C., and Yatani, K. Mixed-initiative approaches to global editing in slideware. In *Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems*, CHI '15, ACM (2015), 3503–3512.

11. Fernquist, J., Shoemaker, G., and Booth, K. S. Oh Snap - helping users align digital objects on touch interfaces. In *Proceedings of the 13th IFIP TC 13 International Conference on Human-computer Interaction - Volume Part III*, INTERACT'11, Springer-Verlag (2011), 338–355.

12. Frisch, M., Langner, R., and Dachselt, R. Neat: A set of flexible tools and gestures for layout tasks on interactive displays. In *Proceedings of the ACM International Conference on Interactive Tabletops and Surfaces*, ITS '11, ACM (2011), 1–10.

13. Garcia, J., Tsandilas, T., Agon, C., and Mackay, W. E. Structured observation with polyphony: A multifaceted tool for studying music composition. In *Proceedings of the 2014 Conference on Designing Interactive Systems*, DIS '14, ACM (2014), 199–208.

14. Glaser, B. G., and Strauss, A. L. *The Discovery of Grounded Theory: Strategies for Qualitative Research*. Aldine de Gruyter, 1967.

15. Gleicher, M. Briar: A constraint-based drawing program. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '92, ACM (1992), 661–662.

16. Gleicher, M. Integrating constraints and direct manipulation. In *Proceedings of the 1992 Symposium on Interactive 3D Graphics*, I3D '92, ACM (1992), 171–174.

17. Igarashi, T., Matsuoka, S., Kawachiya, S., and Tanaka, H. Interactive beautification: A technique for rapid geometric design. In *Proceedings of the 10th Annual ACM Symposium on User Interface Software and Technology*, UIST '97, ACM (1997), 105–114.

18. Janecek, P., Ratzer, A. V., and Mackay, W. E. Redesigning design/cpn: Integrating interaction and petri nets in use. In *Proceedings of Second Workshop on Practical Use of Coloured Petri Nets and Design/CPN,Aarhus, Denmark*, Citeseer (1999), 119–131.

19. Kurlander, D., and Feiner, S. Inferring constraints from multiple snapshots. *ACM Trans. Graph. 12*, 4 (Oct. 1993), 277–304.

20. Mackay, W. E. Video artifacts for design: Bridging the gap between abstraction and detail. In *Proceedings of DIS 2000, Designing Interactive Systems*, ACM (August 2000), 72–82.

21. Mackay, W. E. Using video to support interaction design. In *CHI '02 Extended Abstracts on Human Factors in Computing Systems, DVD Tutorial*, CHI EA '02, ACM (2002).

22. Masui, T. HyperSnapping. In *Proceedings of the IEEE 2001 Symposia on Human Centric Computing Languages and Environments (HCC'01)*, HCC '01, IEEE Computer Society (2001), 188–.

23. Nelson, G. Juno, a constraint-based graphics system. *SIGGRAPH Comput. Graph. 19*, 3 (July 1985), 235–243.

24. O'Donovan, P., Agarwala, A., and Hertzmann, A. DesignScape: Design with interactive layout suggestions. In *Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems*, CHI '15, ACM (2015), 1221–1224.

25. Raisamo, R. An alternative way of drawing. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '99, ACM (1999), 175–182.

26. Ryall, K., Marks, J., and Shieber, S. An interactive constraint-based system for drawing graphs. In *Proceedings of the 10th Annual ACM Symposium on User Interface Software and Technology*, UIST '97, ACM (1997), 97–104.

27. Sutherland, I. E. Sketch Pad: a man-machine graphical communication system. In *Proceedings of the SHARE Design Automation Workshop*, DAC '64, ACM (1964), 6.329–6.346.

28. Van Wyk, C. J. A high-level language for specifying pictures. *ACM Trans. Graph. 1*, 2 (Apr. 1982), 163–182.

29. Wigdor, D., Benko, H., Pella, J., Lombardo, J., and Williams, S. Rock & Rails: Extending multi-touch interactions with shape gestures to enable precise spatial manipulations. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '11, ACM (2011), 1581–1590.

30. Wybrow, M., Marriott, K., Mciver, L., and Stuckey, P. J. Comparing usability of one-way and multi-way constraints for diagram editing. *TOCHI 14*, 4 (Jan. 2008), 19:1–19:38.

31. Xia, H., Araujo, B., Grossman, T., and Wigdor, D. Object-oriented drawing. In *Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems*, CHI '16, ACM (New York, NY, USA, 2016), 4610–4621.

32. Xu, P., Fu, H., Igarashi, T., and Tai, C.-L. Global beautification of layouts with interactive ambiguity resolution. In *Proceedings of the 27th Annual ACM Symposium on User Interface Software and Technology*, UIST '14, ACM (2014), 243–252.

33. Xu, P., Fu, H., Tai, C.-L., and Igarashi, T. GACA: Group-aware command-based arrangement of graphic elements. In *Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems*, CHI '15, ACM (2015), 2787–2795.

34. Zeleznik, R., Bragdon, A., chi Liu, C., and Forsberg, A. Lineogrammer: creating diagrams by drawing. In *In UIST '08: Proceedings of the 21st annual ACM symposium on User interface software and technology* (2008), 161–170.

APPENDIX

The following pseudocode describes how the layout is updated when a guideline is moved by the user.

```
StickyLine.Dragged(Point p) {
    update my location
    for each attached shape s {
        set s as dragged by me
        for each stickyline sl
            sl.snap(s);  // try to snap shape
    }
}

StickyLine.snap(Shape shape) {
    if (shape is attached to me) {
        point = get shape snap point in shape
        dist = distance between me and point
        snapped = true
        if (shape or its bounding box is being
            dragged away from me)
        or (shape belongs to a line being
             dragged with higher priority)
        or (I am being dragged
             and shape belongs to another line
                with higher priority
             and the new position would break
                the relationship) {
                    detach shape from me
                    snapped = false
            }
    } else { // should I attach shape?
        snapped = false
        if (shape is not in an exclusive line)  {
            point = closest snap point in shape
            dist = distance between me & point
            if  (shape is close enough)
            or (shape belongs to a line being
                 dragged and close enough)
                    snapped=true;
        }
    }

    if (snapped) {
        // shape must be added to me
        closer = my closest point
                    to the shape snap point
        move shape according to
             attachment type and tweaks
        add (shape, closer) to my list of
                              attached shape
    }

    if (shape was added or removed
       and distribution is on)
        this.redistribute();
}
```